SF298 MASTER COPY KEEP THIS COPY FOR REPRODUCTION PURPOSES

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*

| | | |
|---|---|---|
| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED<br>Reprint |

| | |
|---|---|
| 4. TITLE AND SUBTITLE<br>Title shown on Reprint | 5. FUNDING NUMBERS<br>ARO MIPR 156-94 |
| 6. AUTHOR(S)<br>Author(s) listed on Reprint | |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)<br>Naval Postgraduate School<br>Monterey, CA 93943 | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)<br>U. S. Army Research Office<br>P. O. Box 12211<br>Research Triangle Park, NC 27709-2211 | 10. SPONSORING/MONITORING AGENCY REPORT NUMBER<br>ARO 30989.21-MA |

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

| | |
|---|---|
| 12a. DISTRIBUTION/AVAILABILITY STATEMENT<br>Approved for public release; distribution unlimited. | 12b. DISTRIBUTION CODE |

13. ABSTRACT *(Maximum 200 words)*

ABSTRACT ON REPRINT

DTIC QUALITY INSPECTED 2

| | |
|---|---|
| 14. SUBJECT TERMS | 15. NUMBER OF PAGES |
| | 16. PRICE CODE |

| | | | |
|---|---|---|---|
| 17. SECURITY CLASSIFICATION OF REPORT<br>UNCLASSIFIED | 18. SECURITY CLASSIFICATION OF THIS PAGE<br>UNCLASSIFIED | 19. SECURITY CLASSIFICATION OF ABSTRACT<br>UNCLASSIFIED | 20. LIMITATION OF ABSTRACT<br>UL |

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Presenters: David A. Dampier, Ronald B. Byrnes, Mark R. Kindl, Luqi
Title: Rapid Prototyping Of Army Embedded Software Systems
Track: Software Engineering Environments
Day: Tuesday, 11 April 1995
Keywords: Rapid Prototyping, Software Development, Embedded Systems, Army Software

# Rapid Prototyping of Army Embedded Software Systems

David A. Dampier

BB4Sdamp.ppt MAC (1)

## Abstract

The Software Technology Branch of the Army Research Laboratory has established a testbed to evaluate the usefulness of rapid prototyping technology for developing embedded real_time software for Army systems. It is still early to make conclusions, but preliminary efforts look promising. Current efforts and future proposed efforts are outlined in this presentation.

Keywords: Rapid Prototyping, Software Development, Embedded Systems, Army Software

## 1. Introduction

Computer-aided rapid prototyping is a software evolution methodology that allows the rapid development of software prototypes with the goal of achieving a validated set of requirements more quickly than under current practice. In more traditional methods of software development, requirements engineering is accomplished by a skilled software engineer after receiving the customer's initial vision for the system. Generally, the customer does not understand enough about the software requirements, and the engineer doesn't understand enough about the customer's problem. The results are incomplete or inaccurate requirements which may lead to software that doesn't completely satisfy the need. Alternatively, periodic design reviews may reveal the problem, but subsequent design changes may result in schedule or cost overruns. Embedded software systems are even more susceptible to these problems, as their functionality is usually hidden behind hardware interfaces. Failure of embedded systems due to software design flaws is generally more serious, as their failure is more likely to result in loss of life, limb, or valuable property.

Computer-aided rapid prototyping provides a method through which the customer and the designer collaborate throughout the design process; thus, the problems associated with requirements analysis are addressed. As shown in Figure 1, an initial set of requirements is provided to the prototype designer who develops an initial prototype. This initial prototype is then demonstrated to the customer for evaluation and feedback. Deficiencies noted by the customer are used by the designer to adjust the requirements and redesign the prototype. The prototype is then redemonstrated to the customer for additional feedback. This iterative process continues until the customer is satisfied with the performance of the prototype and a validated set of requirements is produced. The Ada code generated for the final prototype, including atomic objects created or retrieved from a software base, may then be used as the basis for building the production system. In military software development organizations, this technology can be used by software engineers to quickly produce a set of requirements that satisfies the customer's needs. The resultant system can then be completed by the organization or the prototype and validated requirements can be turned over to a contractor. Either option will provide for more rapid completion of military software with enhanced assurance that the system will accurately satisfy the military requirement.

Although this technology may be used for information systems, the real power is realized in the development of real-time embedded systems. Example software systems that could benefit from this technology include flight control software for aircraft, missiles, and autonomous aviation vehicles, command and control systems, robot control software, and radar monitoring software.

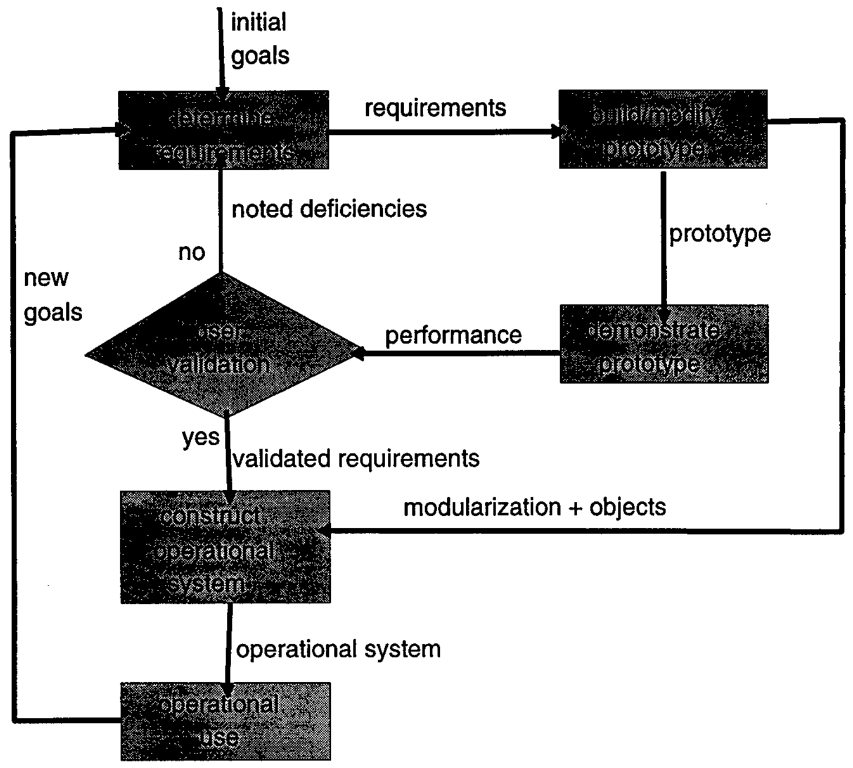

*Figure 1: Computer-Aided Rapid Prototyping Paradigm [4]*

According to [8], prototyping has three main benefits: it improves communication, reduces risk, and is the most feasible way to validate specifications. Communication is improved through demonstration of the prototype to the customer, thus enabling more effective collaboration between the customer and the user and helping expose unstated assumptions from both the customer's and the designer's point of view. Prototyping reduces risk by uncovering the proposed design's unknown properties and providing a basis from which to evaluate alternative designs. Since customer and designer are collaborating on the development of the prototype, it is more likely they will interpret the specification in the same way. This in turn serves to validate the specification during development.

## 2 Rapid Prototyping at the Army Research Laboratory

At the Software Technology Branch of the Army Research Laboratory, we have established a rapid prototyping laboratory to evaluate the usefulness of this technology for developing embedded real-time software for Army weapons systems. The laboratory consists of a SPARC machine hosting the prototyping system along with all required supporting software connected to our network. Additionally, three other SPARC machines are configured to run the software through the network.

One of our goals is to obtain requirements for Army embedded systems such as flight control software for Army aircraft or missiles, autonomous vehicle control software, or command and control systems. These requirements will be used to develop software prototypes to show that rapid prototyping technology can be used to satisfy Army software requirements for embedded real-time software. Once the Army software community is convinced that the technology is capable to meet their needs, the system can be released to Army (and DoD) software developers for use on current systems under development.

An additional, recently proposed goal includes modelling the software development process over a common system evolution record [10]. Using our prototyping laboratory, we will model each part of the software development process and automate each process where possible. A common data structure for tracking all information about a software system from initial goals identification to system retirement will also be modelled using the system. This data structure will be similar to that proposed by Salasin in [10].

## 3 Computer-Aided Prototyping System

The Computer-Aided Prototyping System (CAPS) [7], developed at the Naval Postgraduate School, will be the prototyping environment used to test this technology. CAPS has been used to develop numerous software prototypes including missile system simulators, command and control systems, robot control software, and automated monitoring software [6, 9]. CAPS, as depicted in Figure 2, provides the capability for the designer to design a prototype in a high level specification language, the Prototype System Description Language (PSDL) [6]. PSDL prototypes are executable specifications consisting of sets of operators and abstract data types. Each element of the prototype, whether operator or abstract data type, has a PSDL specification and an implementation written either in PSDL for composite implementations or Ada for atomic implementations. PSDL implementations are enhanced data flow diagrams containing operators and data streams along with control and timing constraints to specify control flow and real-time requirements. In addition, CAPS contains a number of tools to assist the prototype designer. These tools are grouped according to functionality into four basic groups: Editors, Execution Support tools, Project Control tools, and a Software Database.

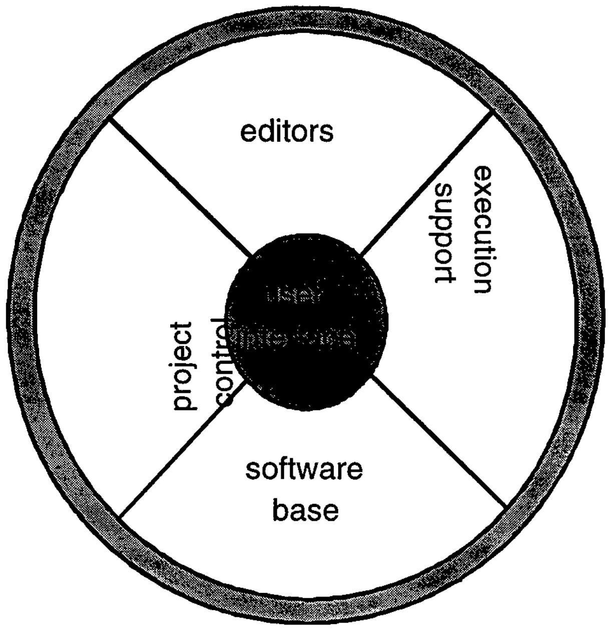

*Figure 2: Computer-Aided Prototyping System.*

## 3.1 Editors

There are three editors used in the CAPS system: a PSDL Editor consisting of both a syntax-directed editor and a graphic editor; a syntax-directed editor for editing Ada code; and an interface editor for building graphical user interfaces. The PSDL editors are linked so that changes made in one are automatically reflected in the other. This allows the software designer to build a prototype outside the CAPS environment and subsequently use it as input to the system.

## 3.2 Execution Support Tools

The execution support tools in the CAPS environment consist of an expander, a translator, and a scheduler. The expander is used to expand the top level PSDL implementation for the prototype by replacing each of the operators with PSDL implementations by their subgraphs. It also moves all of the control constraint definitions to the top level graph. The translator generates an Ada package containing instantiations of the data streams and driver procedures for each of the atomic operators. The scheduler attempts to find a static schedule for all of the time-critical operators. If a feasible schedule is found, then the scheduler produces an Ada task that calls each of the driver procedures for the time-critical operators. Once the static schedule is completed, a dynamic schedule task is generated for all of the non-time-critical operators. The dynamic schedule invokes the non-time critical operators when the processor is not engaged in executing a time-critical operator.

## 3.3 Project Control Tools

The project control tools consist of an evolution control system and a change-merge tool. The evolution control system manages the configuration control of the prototype and schedules design tasks within the design team [1]. The change-merge tool is used to combine independently developed variations of a prototype when different design tasks have been assigned to different designers [2,3,4,5].

Building a prototype in CAPS is accomplished as follows. First, the designer draws the graphic implementation of the prototype using the graphic editor. The graphic editor then automatically provides the skeleton PSDL code for the prototype and propagates any inherited timing and control constraints. Using the syntax directed editor, the designer modifies the skeleton code created by the graphic editor to complete the PSDL description of the prototype. Next, the translator is used to produce an Ada package for instantiating the data streams, reading from and writing to the data streams, and executing the atomic operators. The translator generates driver procedures for each of the atomic operators that provide standard interfaces between the atomic components Ada implementation and the generated schedules. Following this, the static scheduler attempts to create a schedule for all of the time-critical operators. If a feasible schedule is found, a schedule is produced in the form of an Ada task that calls each of the driver procedures for the time-critical operators. Finally, the prototype is compiled, loaded, and executed for the user.

## 3.4 Software Base

During the development of the prototype, the atomic level operators are implemented in a high-level programming language (Ada for military systems). The atomic operator implementations can either be written by the design team or retrieved from a reusable software repository called the software base. This repository contains reusable software components written in a high-level programming language, along with their PSDL specifications. A specification is used during the retrieval process to identify a component and its capabilities. To retrieve one of these reusable components, the designer specifies the desired functionality desired using axioms. The software base is then searched for a component that best matches that functionality. In some cases, there may be more than one component found that provides the desired functionality. In that case, all candidate components would be provided to the designer for consideration.

# 4 Examples of Prototypes Developed using CAPS

The following sections show two examples of real-time embedded prototypes developed using the CAPS software. Although these examples were not built to precise military system requirements, they do demonstrate the possibilities available for developing this type of software. In the example graphs, the bubbles or vertices are operators and the edges are data streams. Some of the operators have MAXIMUM EXECUTION TIMES shown in the graph above the bubble. This is a timing constraint that tells the scheduler that the operator will complete execution within the time shown. Each of the operators in the graph can be implemented in a high-level programming language like Ada or PSDL. If the operator is further decomposed into a PSDL implementation, the implementation would also take the form of a graph.

## 4.1 Patriot Missile Control System

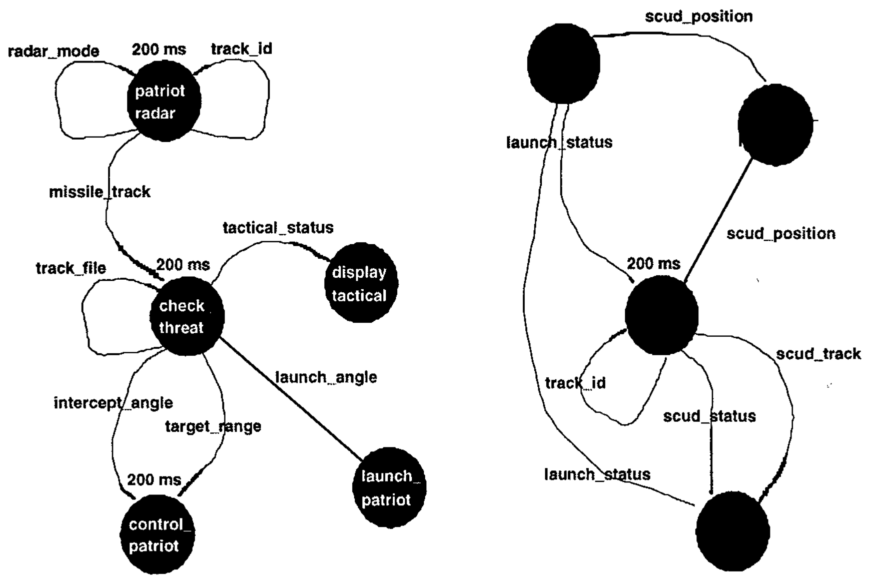

*Figure 3: Prototype for a Patriot Simulator.*

The Patriot prototype shown in Figure 3 was developed shortly after the Gulf War by an instructor and students at the Naval Postgraduate School. It contains a simulated SCUD missile and a Patriot interceptor. The prototype demonstrates the ability of the CAPS software to generate code for an autonomous patriot system. Using a process of iterative refinement, this prototype took two weeks to construct. To execute the prototype, two inputs are necessary: the distance of the SCUD launcher from the friendly border and the distance from the SCUD launcher to the target. If the target is in friendly territory, the Patriot missile will intercept the SCUD before it detonates on target. If the target is in enemy territory, the Patriot system will track the SCUD but will not intercept it.

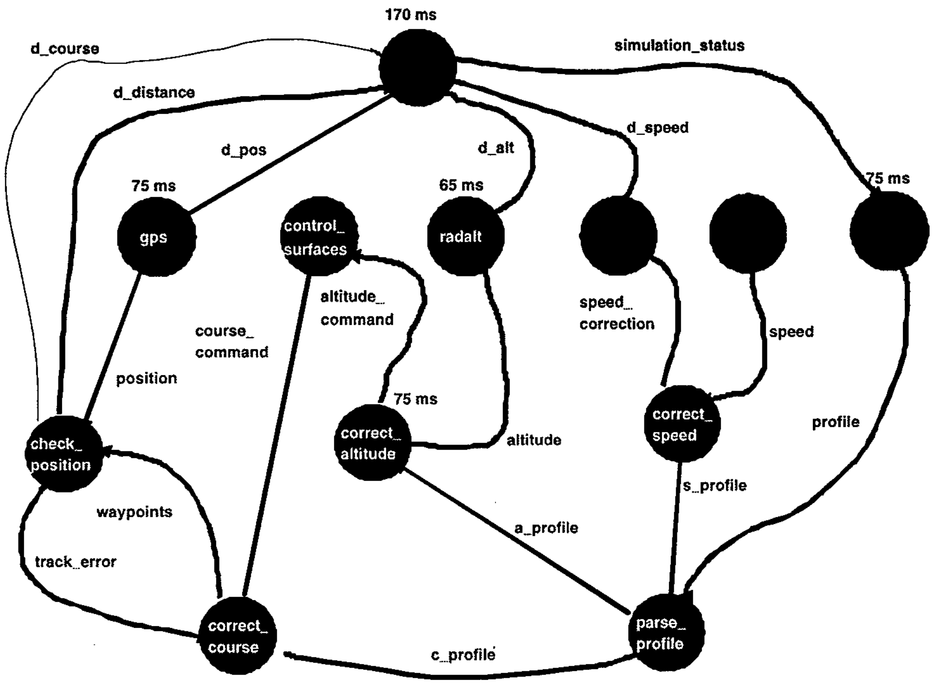

*Figure 4: Prototype for a Tomahawk Missile Simulator.*

## 4.2 Tomahawk Missile Simulator

The prototype shown in Figure 4 is a Tomahawk Missile Simulator. This prototype was constructed by Lieutenant Jim Brockett, U.S. Navy, a Ph.D. Candidate at the Naval Postgraduate School. This prototype simulates a Tomahawk Cruise Missile being fired from a Navy ship and following a predetermined cruise path. The coordinates of checkpoints along the path are input by the user. Once launched, the missile follows the flight path and explodes on target. This prototype demonstrates the ability of CAPS to generate guidance control software.

## 5. Conclusion

We have introduced the rapid prototyping effort being undertaken at the Software Technology Branch of the Army Research Laboratory together with the capabilities of the CAPS software. It is our purpose to generate interest in our project and to solicit from interested agencies real-world requirements for embedded real-time software. This will provide us an opportunity to further demonstrate the usefulness of this system to Army customers.

## 6. References

1. Badr, S., A Model And Algorithm For An Evolution Control System, Ph.D. Dissertation, U.S. Naval Postgraduate School, Monterey, CA, December 1993.

2. Dampier, D., A Model for Merging Different Versions of a PSDL Program, Master's Thesis, U.S. Naval Postgraduate School, Monterey, CA, June 1990.

3. Dampier, D., Luqi, Berzins, V., "Automated Merging of Software Prototypes", Journal Of Systems Integration, Kluwer Academic Publishers, March 1994.

4. Dampier, D., A Formal Method for Semantics-Based Change-Merging of Software Prototypes, Ph.D. Dissertation, U.S. Naval Postgraduate School, Monterey, CA, June 1994.

5. Dampier, D., Byrnes, R., and Kindl, M., "Computer-Aided Maintenance for Embedded Real-Time Software", Proceedings of the 19th Army Science Conference, Orlando, FL, June 1994.

6. Luqi, Berzins, V. and Yeh, R., "A Prototyping Language for Real-Time Software", IEEE Transactions on Software Engineering, October 1988, pp. 1409-1423.

7. Luqi, "Software Evolution Through Rapid Prototyping", IEEE Computer, May 1989.

8. Luqi and Royce, W., "Status Report: Computer-Aided Prototyping", IEEE Software, November 1991, pp. 77-81.

9. Luqi, "Computer-Aided Prototyping for a Command and Control System using CAPS", IEEE Software, January 1992, pp. 56-67.

10. Salasin, J., "The System Evolution Record", Proceedings of the First Reengineering Workshop, Santa Barbara I, 21 - 25 September 1992, pp. 4-8 - 4-17.

## DAVID A. DAMPIER, Ph.D.

CPT David A. Dampier is a Systems Automation Engineer with the Software Technology Branch, Computational Sciences and Technology Division, Advanced Computational and Information Sciences Directorate, Army Research Laboratory located at the Georgia Institute of Technology in Atlanta, Georgia. His research interests are in software engineering, software prototyping, software evolution, and formal methods for configuration control and change-merging.

CPT Dampier's previous assignments include Missile Maintenance Company Commander and Division Repair Parts Accountable Officer with the 24th Infantry Division, and Communications-Electronic Materiel Management Officer with the 2d Infantry Division.

CPT Dampier was awarded his B.S. in Mathematics from the University of Texas at El Paso in 1984, and his M.S. and Ph.D. degrees in Computer Science from the Naval Postgraduate School in 1990 and 1994 respectively.